**Task 3a:**

***Define the following: Task, Pipelining, Shared Memory, Communications, Synchronization. (in your own words)***

Task - A program or set of instructions that is executed by the processor to do a certain thing

Pipelining - A type of parallel computation where the work is broken down into smaller steps

Shared Memory - A type of architecture where all of the processors share access to the same physical memory

Communications - The exchange of data between different parallel processes

Synchronization - Deals with the coordination of tasks in parallel computation

***Classify parallel computers based on Flynn's taxonomy. Briefly describe every one of them.***

Single Instruction, Single Data - Only one instruction stream is being acted on by the CPU during any one clock cycle and only one data stream is being used as input during any one clock cycle.

Single Instruction, Multiple Data - All processing units execute the same instruction at any given clock cycle and each processing unit can operate on a different data element.

Multiple Instruction, Single Data - Each processing unit operates on the data independently via separate instruction streams and a single data stream is fed into multiple processing units.

Multiple Instruction, Multiple Data - Every processor may be executing a different instruction stream and every processor may be working with a different data stream.

***What are the Parallel Programming Models?***

Shared memory, threads, distributed memory/message passing, data parallel, hybrid, single program multiple data, multiple program multiple data.

***List and briefly describe the types of Parallel Computer Memory Architectures. What type is used by OpenMP and why?***

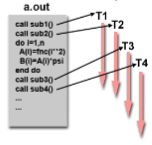
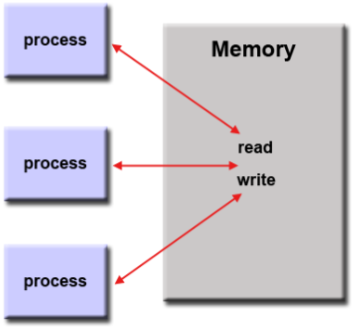
Uniform Memory Access - Identical processors, equal access and access times to memory.

Non-Uniform Memory Access - One SMP can directly can directly access of another SMP, not all processors have equal access time to all memories.

OpenMP can use both types of architectures so that the code can be as portable as possible.

***Compare Shared Memory Model with Threads Model? (in your own words and show pictures)***

In the Shared Memory Model, processes and tasks share a common space in the memory which they use read and write at the same time, while in the Threads Model, work is split up into several smaller threads that are executed at the same time.



***What is Parallel Programming? (in your own words)***

Parallel programming is a way of programming in which multiple cores of the system are used to perform multiple tasks at the same time.

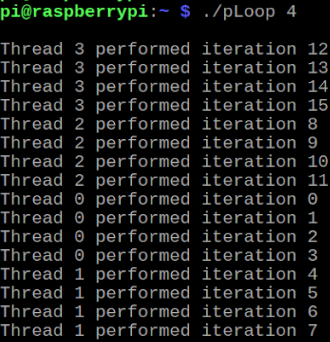
***What is system on chip (SoC)? Does Raspberry PI use system on SoC?***

System on chip is when the CPU, RAM, and GPU are all squeezed into one component, usually a single silicon chip. The Raspberry Pi utilizes the SoC.

***Explain what the advantages are of having a System on a Chip rather than separate CPU, GPU and RAM components.***

A CPU is slightly more compact than a SoC, but the SoC contains a lot more functionality. A SoC also consumes much less power than a CPU, and also much cheaper. The biggest advantage of a CPU over a SoC is that it is much more flexible, in that new components can be put in at any time.

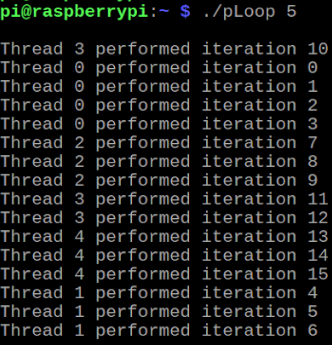
**Task 3b:**



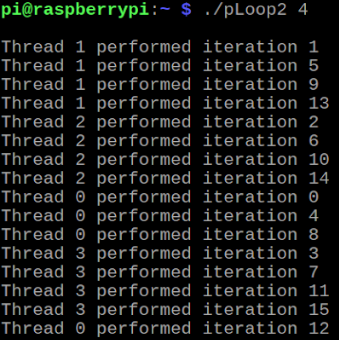
This was the output after running the first code. In this iteration of the code, the number of iterations evenly divides the number of threads, and so each thread performs an equal number of iterations.



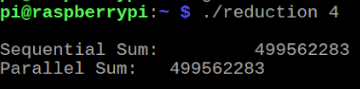
When the the number of iterations does not evenly divide the number of threads, some threads will perform less iterations than others. In this case, Thread 0 performed 6 iterations, while Threads 1 and 2 both performed 5.



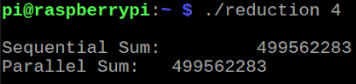
When performed with a different uneven number, Thread 0 is again the thread performing the most iterations.



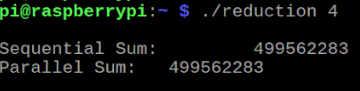
With the second code, the difference is that the assignment of iterations to threads is done in rounds (as in Thread 0 performs iteration 0, 1 performs 1, 2 performs 2, 3 performs, then back around to Thread 0 which performs iteration 4, and it continues in that manner) instead of each thread performing a specific range of iterations.



This is the output after the sum code is first run.



After the first comment mark is removed, the output is still the same.



After all of the comment marks are removed, the output still remains the same. I checked all of the code several times to make sure that I had it right, because this was not the result that I was expecting.